Object-Oriented Programming Language

06/01/2023

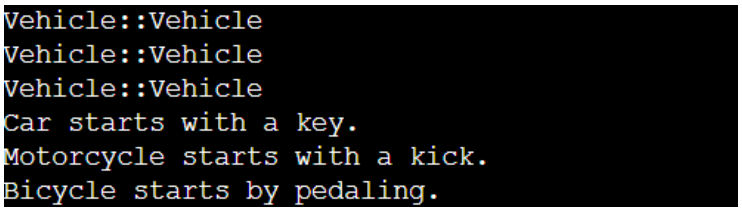
Due:06/05/2023 11:59p.m.

Lab Assignment 13

1. Create a base class Vehicle with a function start() and create 3 sub-classes: Car, Bicycle and Motorcycle. The following is the main code you should not change.

```
int main() {
    Vehicle* vehicle1 = new Car();
    Vehicle* vehicle2 = new Motorcycle();
    Vehicle* vehicle3 = new Bicycle();

    vehicle1->start();
    vehicle2->start();
    vehicle3->start();

    delete vehicle1;
    delete vehicle2;
    delete vehicle3;
}
```

Output:

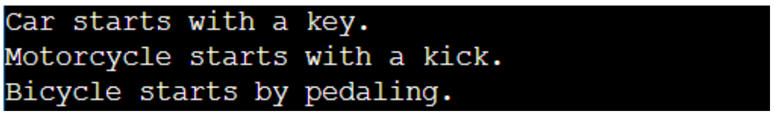


```
Vehicle::Vehicle
Vehicle::Vehicle
Vehicle::Vehicle
Car starts with a key.
Motorcycle starts with a kick.
Bicycle starts by pedaling.
```

2. Extending the previous problem, set start() to be a pure virtual function. The following is the main code you should not change.

```
int main() {
    Vehicle* vehicle1 = new Car();
    Vehicle* vehicle2 = new Motorcycle();
    Vehicle* vehicle3 = new Bicycle();
    vehicle1->start();
    vehicle2->start();
    vehicle3->start();

    delete vehicle1;
    delete vehicle2;
    delete vehicle3;
}
```

Output:



```
Car starts with a key.
Motorcycle starts with a kick.
Bicycle starts by pedaling.
```

3. Define a base class called Shape which has two pure virtual functions: double getArea() and string getClassName(). Implement three derived classes: Rectangle, Circle, and Triangle.

   The **Rectangle** class should have:
   - A double, len, representing the length of the rectangle; and
   - A double, wid, representing the width of the rectangle.

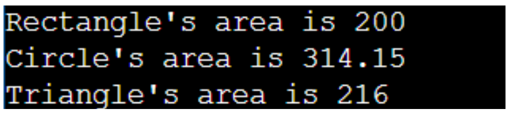
   The **Circle** class should have:
   - A double, rad, representing the radius of the circle.

   The **Triangle** class has:
   - 3 Doubles, a, b, and c, representing the lengths of the three sides of the triangle.

   All of the derived classes override getArea() and getClassName(). The following is the main code you should not change.

```
int main() {
    vector<Shape*> vs;
    Rectangle r(10,20);
    Circle c(10);
    Triangle t(18,30,24);
    vs.push_back(&r);
    vs.push_back(&c);
    vs.push_back(&t);
    for(auto s : vs)
        cout << s;
}
```

Output:

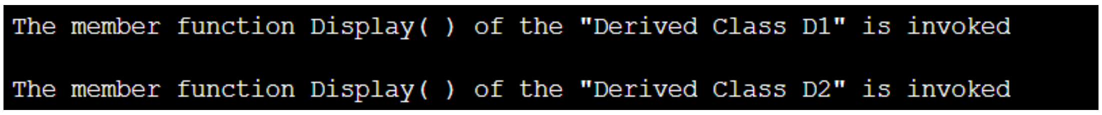


```
Rectangle's area is 200
Circle's area is 314.15
Triangle's area is 216
```

4. Write a C++ Program to demonstrate Run time polymorphism.
   The following is the main code you should not change.

```
int main()
{
        B* base_ptr ;
        D1 der1_obj ;
        base_ptr = &der1_obj ;
        base_ptr->Display( );
        D2 der2_obj ;
        base_ptr = &der2_obj ;
        base_ptr->Display( );
}
```

   Output should look like this



```
The member function Display( ) of the "Derived Class D1" is invoked

The member function Display( ) of the "Derived Class D2" is invoked
```

5. Following problem 3, class Shape is also an ABC, but change getArea() into void function which prints the area directly.

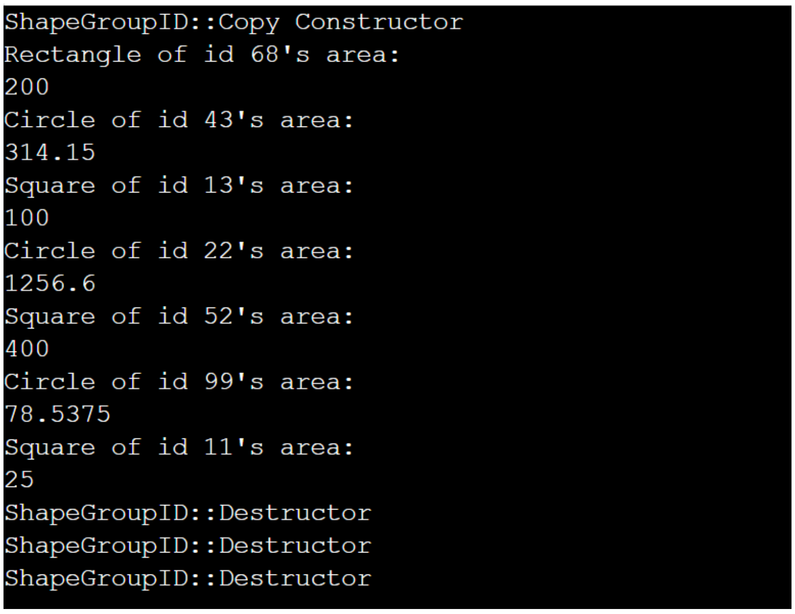
   Now design a new class ShapeGroupID which inherits from class Shape representing a group of shapes. It should have a vector to store shape objects and another vector to store corresponding IDs. Implement the necessary functions to insert shapes into the group and retrieve their areas and class names.

   The structure of ShapeGroupID is listed below, and the following main code should not be change.

```
class ShapeGroupID : public Shape {
protected:
    vector<Shape*> elems;
    vector<int> IDs;
public:
    ShapeGroupID() = default;

    void getArea() {
       // input your code here
    }
    string getClassName() {
       // input your code here
    }
    void insert(Shape* s) {
        elems.push_back(s);
        IDs.push_back(rand() % 100);
    }
    void insert(ShapeGroupID& sgid) {
            // input your code here
    }
    // copy constructor
    ShapeGroupID(const ShapeGroupID& rhs) {
            // input your code here
    }
    // destructor
    ~ShapeGroupID() {
            // input your code here
    }
};

int main() {
    srand(420);
    ShapeGroupID sgID;

    Rectangle* r1 = new Rectangle(10, 20);
    Circle* c1 = new Circle(10);
    Square* s1 = new Square(10);
```

```
        sgID.insert(r1);
        sgID.insert(c1);
        sgID.insert(s1);

        ShapeGroupID sgID2;
        Circle* c2 = new Circle(5);
        Square* s2 = new Square(5);
        sgID2.insert(c2);
        sgID2.insert(s2);

        ShapeGroupID sgID3(sgID);
        Circle* c3 = new Circle(20);
        Square* s3 = new Square(20);
        sgID3.insert(c3);
        sgID3.insert(s3);
        sgID3.insert(sgID2);

        sgID3.getArea();

        delete r1, c1, s1, c2, s2, c3, s3;
    }
```

Output:



```
ShapeGroupID::Copy Constructor
Rectangle of id 68's area:
200
Circle of id 43's area:
314.15
Square of id 13's area:
100
Circle of id 22's area:
1256.6
Square of id 52's area:
400
Circle of id 99's area:
78.5375
Square of id 11's area:
25
ShapeGroupID::Destructor
ShapeGroupID::Destructor
ShapeGroupID::Destructor
```